6.22 Compression and the Huffman Code

Shannon's Source Coding Theorem (6.52) has additional applications in **data compression**. Here, we have a symbolic-valued signal source, like a computer file or an image, that we want to represent with as few bits as possible. Compression schemes that assign symbols to bit sequences are known as **lossless** if they obey the Source Coding Theorem; they are **lossy** if they use fewer bits than the alphabet's entropy. Using a lossy compression scheme means that you cannot recover a symbolic-valued signal from its compressed version without incurring some error. You might be wondering why anyone would want to intentionally create errors, but lossy compression schemes are frequently used where the efficiency gained in representing the signal outweighs the significance of the errors.

Shannon's Source Coding Theorem states that symbolic-valued signals require **on the average** at least ***H(A)*** number of bits to represent each of its values, which are symbols drawn from the alphabet *A*. In the module on the Source Coding Theorem (Source Coding Theorem (Page 295)) we find that using a so-called **fixed rate** source coder, one that produces a fixed number of bits/symbol, may not be the most efficient way of encoding symbols into bits. What is not discussed there is a procedure for designing an efficient source coder: one **guaranteed** to produce the fewest bits/symbol on the average. That source coder is not unique, and one approach that does achieve that limit is the **Huffman source coding algorithm**.

POINT OF INTEREST: In the early years of information theory, the race was on to be the first to find a **provably** maximally efficient source coding algorithm. The race was won by then MIT graduate student David Huffman in 1954, who worked on the problem as a project in his information theory course. We're pretty sure he received an "A."

- Create a vertical table for the symbols, the best ordering being in decreasing order of probability.
- Form a binary tree to the right of the table. A binary tree always has two branches at each node. Build the tree by merging the two lowest probability symbols at each level, making the probability of the node equal to the sum of the merged nodes' probabilities. If more than two nodes/symbols share the lowest probability at a given level, pick any two; your choice won't affect
$$B\bar{(}A)$$
- At each node, label each of the emanating branches with a binary number. The bit sequence obtained from passing from the tree's root to the symbol is its Huffman code.

Example 6.3

The simple four-symbol alphabet used in the Entropy (Entropy (Page 293)) and Source Coding (Source Coding Theorem (Page 295)) modules has a four-symbol alphabet with the following probabilities,